the globus project
www.globus.org

Replica Management for the Grid

Ann Chervenak

USC Information Sciences Institute

annc@isi.edu

Replica Management in Grids

- Need to manage large scientific computing datasets
  - Terabytes or petabytes shared by researchers around the world
  - Read-only data, “published” by experiments
- Replicate portions of the data set in multiple locations
  - Local control, reduce access times, provide fault tolerance
- Discover replicas and select the best replica for a necessary data transfer

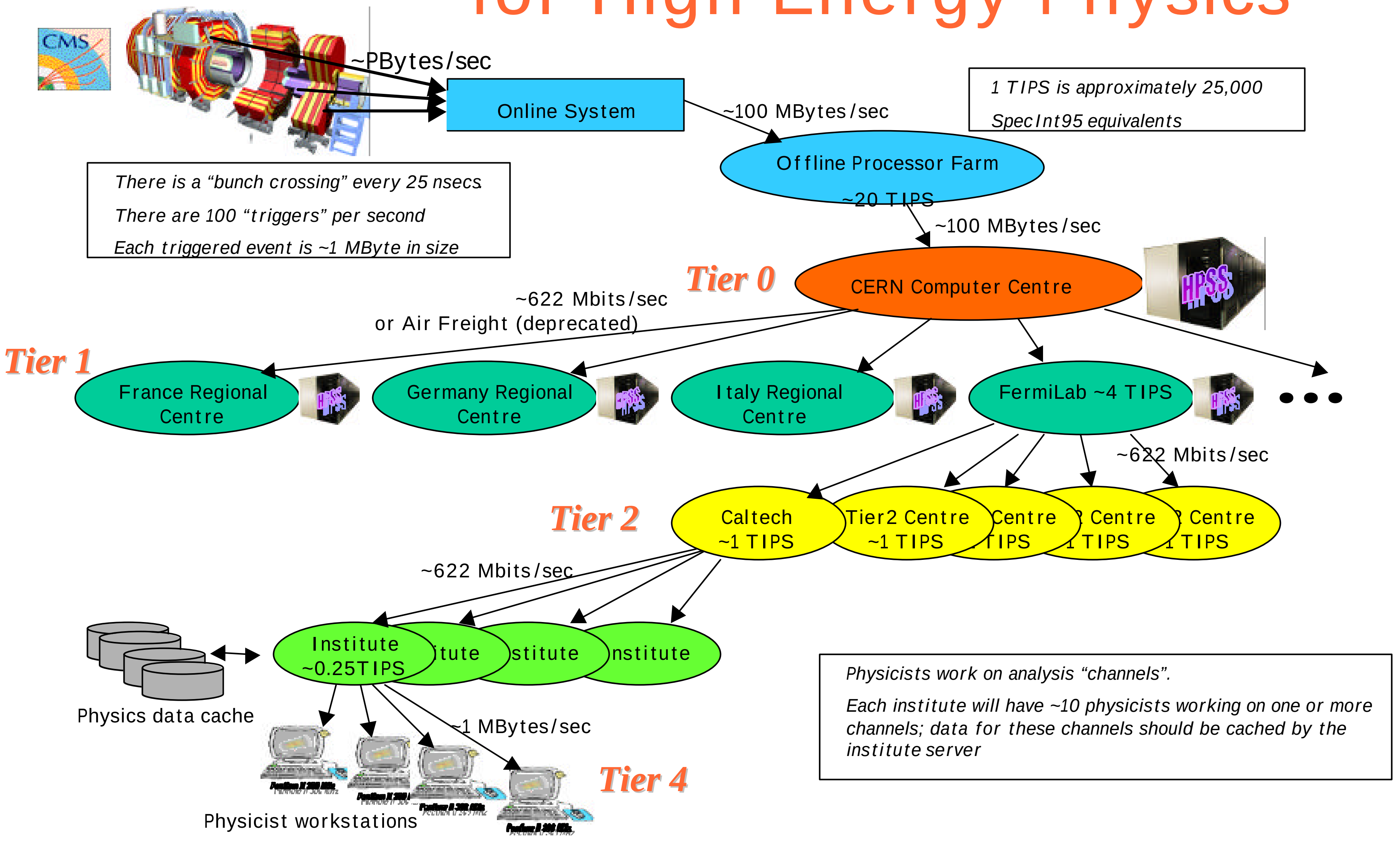
Outline

- Data Intensive Applications: two examples
- Requirements for Data Grids
- A Replica Management System
  - Current implementation: Replica catalog and API for reliable replication
  - Address issues of replica location, file aggregation and reliable replication
- Replica Location Service
  - A flexible design framework
  - Addresses issues of reliability, scalability and performance

Climate Modeling

- Detecting global climate change
- Simulate climate variability over long periods
  - Often 100 years
  - Long-duration computations: 1 month at 100 Gflops
  - Large output files: 10 terabytes
- Compare simulation results to observed variability
- With teraflop computers: requirements will increase by factor of 10 or more
- Need to publish, replicate and share these files with other researchers

Example: Data Replication for High Energy Physics



There is a "bunch crossing" every 25 nsecs.

There are 100 "triggers" per second

Each triggered event is ~1 MByte in size

1 TIPS is approximately 25,000

SpecInt95 equivalents

Physicists work on analysis "channels".

Each institute will have ~10 physicists working on one or more channels; data for these channels should be cached by the institute server

Image courtesy Harvey Newman, Caltech

Data Grid Publication and Replication Requirements

- Terabytes or petabytes of data
  - Often read-only data, “published” by experiments
- Large data storage and computational resources shared by researchers around the world
  - Distinct administrative domains
  - Respect local and global policies governing how resources may be used
- Provide access to:
  - Raw experimental data
  - Simulation and analysis data products

Data Grid Requirements (Cont.)

- Management of data replication
  - Register and query physical copies of files
  - Reliably create and register new replicas
  - Select the best replica for a data transfer
- Security
  - Protect knowledge about existence of data

Fundamental Issues for Replica Management

- ***Location***: finding copies of files
- ***Aggregation***: manage groups of files to improve convenience and scalability
- ***Creation/Reliable replication***: copy files reliably and register them with management system
- ***Scalability***: manage large numbers of files
- ***Performance***: fast response time, large query and update rates
- ***Reliability***: resilient to component failures

The Globus Architecture for Replica Management in Grids

- Managing multiple copies of data in wide area environments
- Identify replica cataloging and reliable replication as two fundamental services
  - Layer on other Grid services: GSI, transport, information service
- Used by higher-level services:
  - Replica selection
  - Automatic creation of new replicas

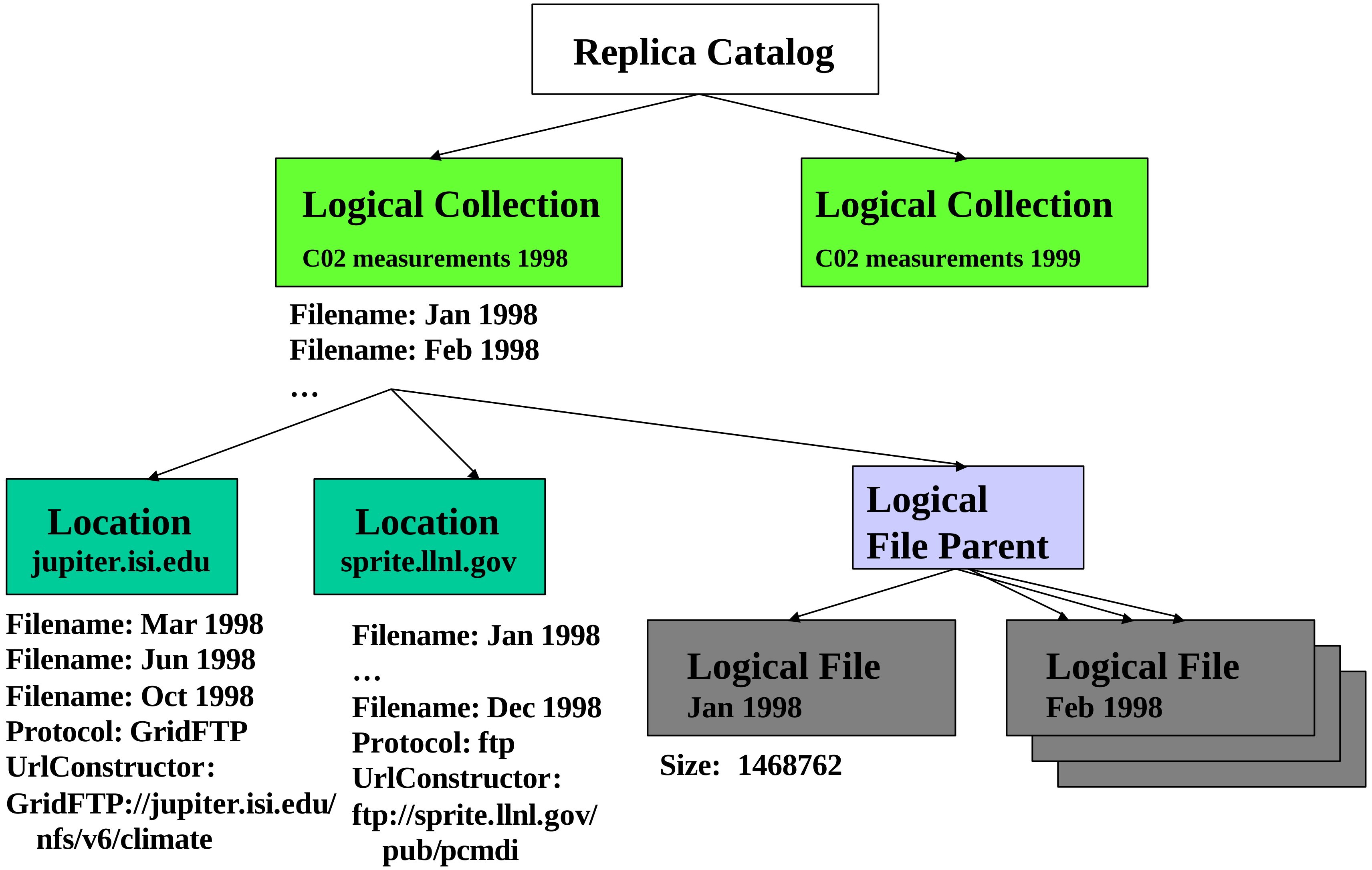
Our Data Model

- Data are organized into ***files***
- Users group files into ***collections***
- A ***replica*** or ***location*** is a subset of a collection stored on a particular physical storage system
- ***Logical file name***: globally unique ID for a file within data grid's namespace
- ***Physical file name***: location of an instance of a file on a particular storage system
- Maintain ***mapping*** between logical names for files and collections and one or more physical locations

The Replica Catalog

- Allows users to register replicas
- Answers queries about existing replicas
- Logical files
  - Entities with globally unique names, may have one or more physical instances
- Logical collection
  - Logical aggregations of groups of files (e.g., simulation timesteps)
- Location entries
  - All information required to map from logical to physical file names (hostname, port number, path…)
  - Location corresponds to one physical storage system

Replica Catalog Structure



Reliable Replication with the Replica Management API

- Reliable replication
  - Combines storage system operations with replica catalog updates
  - Create new replicas reliably
  - Automatically register them in the replica catalog
- Combined operations include:
  - Copy a file from one storage system to another and update the replica catalog
  - Delete a file from storage system and update catalog
- Reliability features
  - If can't complete operation, must **rollback** to previous consistent replica catalog state

Replica Selection

- Built on top of replica management
- Given multiple physical copies of a desired file, want to select the "best" copy for a data transfer
  - Select replica with best estimated performance
- Rely on *information services* that provide dynamic information about grid conditions
  - Storage system latency, bandwidth, load
  - Network latency and bandwidth
  - Authorization
  - User preferences

A Replica Location Service Framework

- Distribute replica management system to avoid single point of failure, performance bottleneck
- Applications may operate at different scales, have different resources and different tolerances to inconsistent RLS information
- We define a *flexible RLS framework*
- Allows users to make tradeoffs among:
  - Consistency, space overhead, reliability, update costs, query costs
- By different combinations of 5 essential elements, the framework supports a variety of RLS designs

Five Essential Elements of a Flexible RLS Framework

1. Reliable Local State

   Maintains consistent information about replicas at a single replica site

   – Updated when files are created or destroyed on local storage system
   – Contains mappings between LFNs & PFNs
   – Answers queries

2. Global State with Relaxed Consistency

   Implement as set of one or more Global Replica Index Nodes

   – Contain some LFN,replica site mappings
   – Accept periodic inputs from sites describing state
   – Answer queries for replicas associated with an LFN

Five Essential Elements of a Flexible RLS Framework

3. Soft State mechanisms for maintaining global state

   Soft state: information that times out and must be periodically refreshed

   – Stale information removed implicitly via timeouts
   – Index node state need not be persistent

4. Compression of State Updates

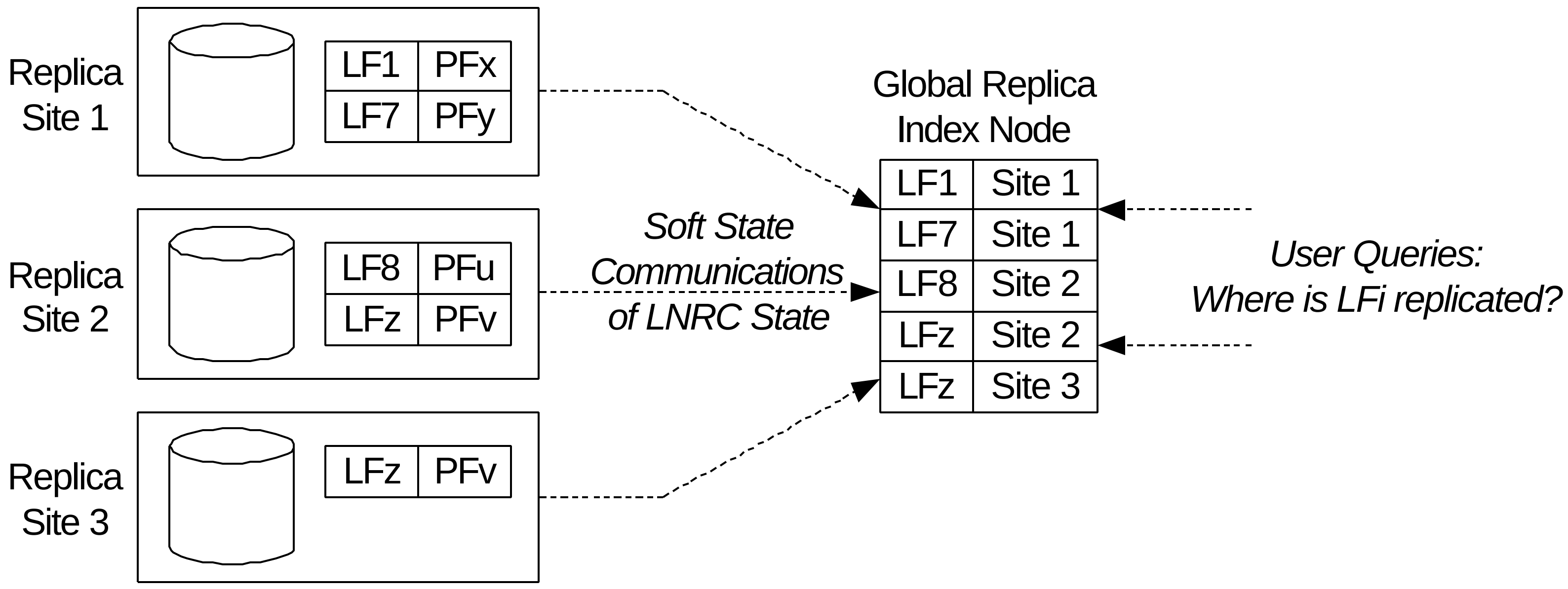
   Optional mechanism for reducing:

   – communication requirements for state updates
   – storage system requirements on GRINs

5. Membership Protocol

   For locating participating replica sites and GRINs

Example 1: A Centralized, Nonredundant Global Index

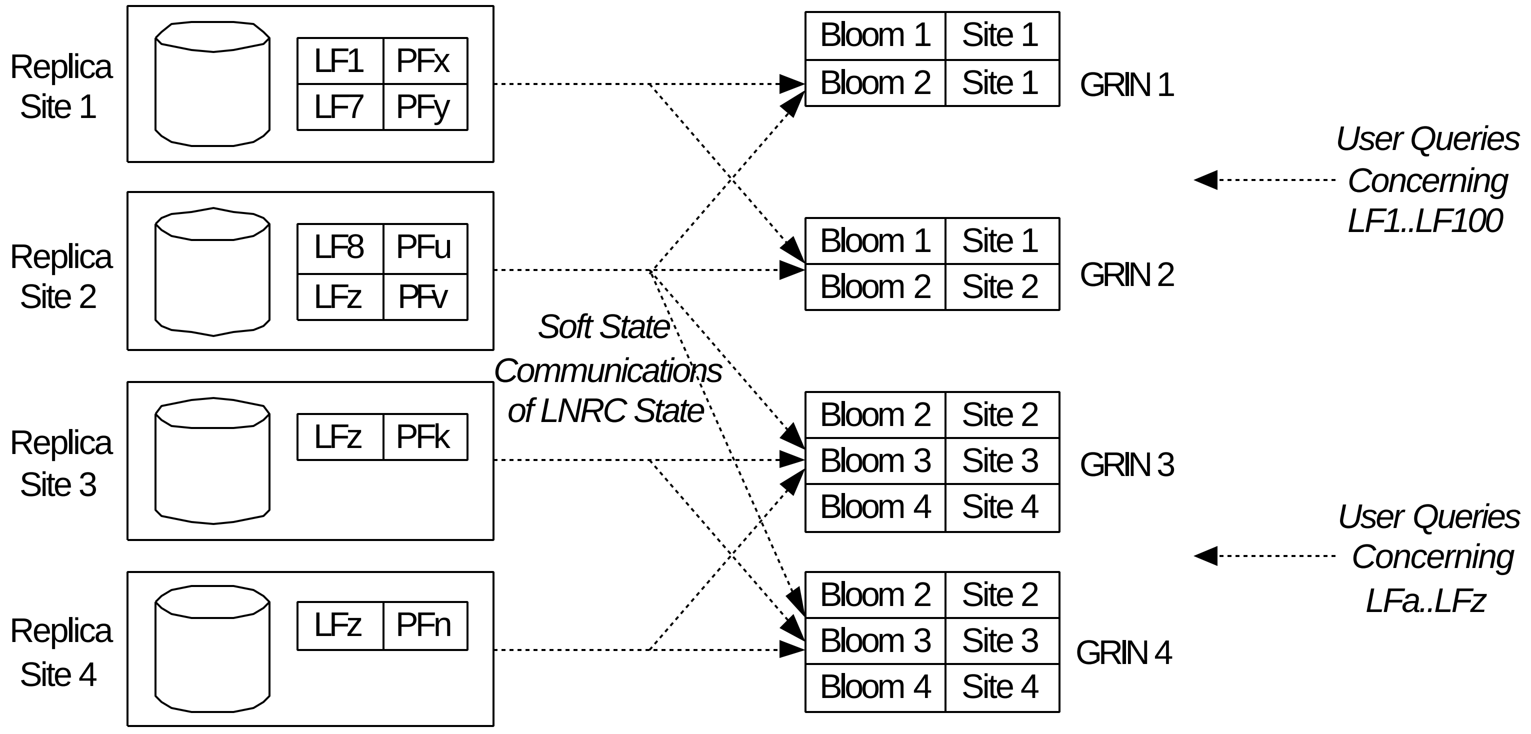


All updates sent to a centralized GRIN

Not scalable: All queries serviced by a single index

Not reliable: Single point of failure

Example 2: An RLS with LFN Partitioning, Redundancy and Bloom Filter Compression



- Updates to specific, redundant GRINs based on LFN
- More scalable, reliable
- Limited storage and communication costs

Summary

- Replica Management is a challenging problem for data-intensive applications
  - Terabytes and petabytes of data
  - Replicated and shared by researchers around the world
- Globus replica management
  - Replica catalog
  - Reliable replication API
- Replica Location Service
  - Framework for flexible design of distributed replica location services
  - Reliable local state, relaxed global state, soft state updates, compression, membership